DISEÑO DE UN PROCESADOR ESPECÍFICO DESCRITO EN VHDL QUE PERMITA CALCULAR CUATRO PARÁMETROS NECESARIOS DENTRO DEL PROCESO DE MIGRACIÓN SÍSMICA.

HENYER RICARDO SEPÚLVEDA SUÁREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2010

DISEÑO DE UN PROCESADOR ESPECÍFICO DESCRITO EN VHDL QUE PERMITA CALCULAR CUATRO PARÁMETROS NECESARIOS DENTRO DEL PROCESO DE MIGRACIÓN SÍSMICA.

Presentado por:

HENYER RICARDO SEPÚLVEDA SUÁREZ

Dirigido por:

Director: MIE (c). CARLOS A. FAJARDO ARIZA
Codirector: MIE (c). SERGIO A. ABREO CARRILLO

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA

2010

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

TITULO:

DISEÑO DE UN PROCESADOR ESPECÍFICO DESCRITO EN VHDL QUE PERMITE CALCULAR CUATRO PARÁMETROS NECESARIOS DENTRO DEL PROCESO DE MIGRACIÓN SÍSMICA[1].

AUTOR

HENYER RICARDO SEPÚLVEDA SUÁREZ[2].

PALABRAS CLAVE

DATAPATH, HDL, MÁQUINA DE ESTADOS FINITA, PUNTO FLOTANTE.

DESCRIPCIÓN

En este artículo se presenta el proceso de diseño de un procesador de propósito específico que calcula el tiempo de viaje, la desaceleración lateral, el coseno del ángulo incidente y el ángulo emergente; cuatro parámetros dentro del proceso de migración sísmica, todos en formato de punto flotante de precisión sencilla. La metodología de diseño empleada tiene como objetivo elaborar módulos de *hardware* que cumplan una función específica, paralelizando la mayor cantidad de operaciones en variables y módulos funcionales para así utilizar la menor cantidad de recursos lógicos. La metodología de diseño consiste en la implementación de un *Datapath,* que se encarga de todo el hardware necesario en el procesador especifico, controlado por una Máquina de Estados Finita (FSM), que permita secuenciar las operaciones, pero para la realización de estos dos módulos se incluye en el flujo de diseño la realización de un Diagrama de Maquina de Estados Algorítmica (ASM) que facilita la construcción de los mismos. Adicionalmente, como en el proceso de diseño se necesita de módulos funcionales para la realización de las operaciones aritméticas y trigonométricas que trabajan en formato de punto flotante, se presenta de forma general el funcionamiento de los módulos funcionales y se enumeran los trabajos y herramientas de donde se tomaron los módulos para el proceso de construcción del procesador específico. Por último, se presenta la validación del diseño, calculando los cuatro parámetros, mediante pruebas de simulación, aprovechando la herramienta de software ISE de Xilinx.

[1]Trabajo de grado.

[2] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: MIE(c) Carlos A. Fajardo. Codirector: MIE(c) Sergio A. Abreo.

ABSTRACT

TITLE:

DESIGN OF A SPECIFIC PROCESSOR DESCRIBED IN VHDL WHICH CAN CALCULATE FOUR PARAMETERS NEEDED IN THE PROCESS OF SEISMIC MIGRATION[3].

AUTHOR

HENYER RICARDO SEPÚLVEDA SUÁREZ[4].

INDEX TERMS

DATAPATH, HDL, FINITE STATE MACHINE, FLOATING POINT.

DESCRIPTION

This paper presents the process the design process of a specific-purpose processor that calculates the travel time, slowness, the cosine of incident angle and the angle emerging, four parameters in the seismic migration process, all on floating point format single-precision. The design methodology used aims to develop hardware modules that meet a specific function, paralleling the largest number of operations in variables and functional modules in order to use the least amount of logic resources. The design methodology consists in the implementation of a Datapath, which is responsible for all the necessary hardware in the specific processor, controlled by a Finite State Machine (FSM), allowing a operations sequence, but for the realization of these modules is included in the design flow diagram conducting an Algorithmic State Machine (ASM) that facilitates the construction of them. Additionally, as in the design process of functional modules required to perform arithmetic and trigonometric operations working in floating point format is presented in a general way the operation of functional modules and lists the tools work and where were taken modules for the process of building the specific processor. Finally, we present the design validation, calculating the four parameters by simulation tests, using the tool Xilinx ISE software.

[3] Work Degree

[4] Faculty of Physical-Mechanical Engineering. School of Engineerings Electrical, Electronic and Telecommunications. Directress: MIE(c) Carlos A. Fajardo. Codirectress: MIE(c) Sergio A. Abreo

Diseño de un procesador específico que permite calcular cuatro parámetros necesarios dentro del proceso de migración sísmica.

Henyer R. Sepúlveda, Carlos A. Fajardo, Sergio A. Abreo

Resumen - En este artículo se presenta el proceso de diseño de un procesador de propósito específico que calcula el tiempo de viaje, la desaceleración lateral, el coseno del ángulo incidente y el ángulo emergente; cuatro parámetros dentro del proceso de migración sísmica, todos en formato de punto flotante de precisión sencilla. La metodología de diseño empleada tiene como objetivo elaborar módulos de hardware que cumplan una función específica, paralelizando la mayor cantidad de operaciones en variables y módulos funcionales para así utilizar la menor cantidad de recursos lógicos. La metodología de diseño consiste en la implementación de un Datapath, que se encarga de todo el hardware necesario en el procesador especifico, controlado por una Máquina de Estados Finita (FSM), que permita secuenciar las operaciones, pero para la realización de estos dos módulos se incluye en el flujo de diseño la realización de un Diagrama de Maquina de Estados Algorítmica (ASM) que facilita la construcción de los mismos. Adicionalmente, como en el proceso de diseño se necesita de módulos funcionales para la realización de las operaciones aritméticas y trigonométricas que trabajan en formato de punto flotante, se presenta de forma general el funcionamiento de los módulos funcionales y se enumeran los trabajos y herramientas de donde se tomaron los módulos para el proceso de construcción del procesador específico. Por último, se presenta la validación del diseño, calculando los cuatro parámetros, mediante pruebas de simulación, aprovechando la herramienta de software ISE de Xilinx.

Palabras clave – Datapath, HDL, Máquina de Estados Finita, Punto Flotante.

Abstract - This paper presents the process the design process of a specific-purpose processor that calculates the travel time, slowness, the cosine of incident angle and the angle emerging, four parameters in the seismic migration process, all on floating point format single-precision. The design methodology used aims to develop hardware modules that meet a specific function, paralleling the largest number of operations in variables and functional modules in order to use the least amount of logic resources. The design methodology consists in the implementation of a Datapath, which is responsible for all the necessary hardware in the specific processor, controlled by a Finite State Machine (FSM), allowing a operations sequence, but for the realization of these modules is included in the design flow diagram conducting an Algorithmic State Machine (ASM) that facilitates the construction of them. Additionally, as in the design process of functional modules required to perform arithmetic and trigonometric operations working in floating point format is presented in a general way the operation of functional modules and lists the tools work and where were taken modules for the process of building the specific processor. Finally, we present the design validation, calculating the four parameters by simulation tests, using the tool Xilinx ISE software.

Index Terms – Datapath, HDL, Finite State Machine, Floating Point.

1. INTRODUCCIÓN

HOY en día, aunque no se haya percibido su presencia, los procesadores de propósito especifico están causando una revolución tecnológica sin precedentes, debido a que estos sistemas están ofreciendo mejores soluciones a las necesidades del ser humano, en sectores cercanos a nuestro entorno como las telecomunicaciones, la medicina y la industria del entretenimiento entre otras.

En el campo de la industria petrolera este tipo de procesadores también se perfilan como una posible fuente de solución a un problema específico, ya que *"en la actualidad, las empresas relacionadas con la industria del petróleo usan clústers de computadores para calcular la migración 2D y 3D"* [1], debido a que los volúmenes de datos a procesar son del orden de los Gigabytes y ocasionan un gran costo computacional, pero estos "*a su vez requieren de grandes sistemas de enfriamiento y su uso acarrea consumos considerables de energía".*

En este orden de ideas, lo que se quiere con este trabajo es, presentar el diseño de un procesador específico que calcula cuatro parámetros necesarios dentro del proceso de migración sísmica, además poder mostrar como por medio de la metodología de diseño expuesta en [2], se puede realizar el diseño de un procesador especifico con un

Henyer R. Sepúlveda pertenece al grupo de investigación en Conectividad y Procesamiento de Señales, Universidad Industrial de Santander, Bucaramanga, Colombia (e-mail: henyer84@hotmail.com).

Carlos A. Fajardo. Director de proyecto. Pertenece al grupo de investigación en Conectividad y Procesamiento de Señales, Universidad Industrial de Santander, Bucaramanga, Colombia (e-mail: cafa_78@yahoo.com).

Sergio A. Abreo. Codirector de Proyecto. Pertenece a los grupos de investigación en Conectividad y Procesamiento de Señales y Petrosísmica, Universidad Industrial de Santander, Bucaramanga, Colombia (e-mail: maepe25@uis.edu.co).

enfoque funcional . El diseño se basa en la realización de un *Diagrama ASM* [2-3] que muestra de manera gráfica y más sencilla el funcionamiento del procesador facilitando la construcción del camino de datos ó *Datapath* y la máquina de estados o Control.

El presente artículo está organizado de la siguiente manera: la sección II presenta de forma general uno de los problemas que se presenta en el proceso de migración sísmica y lo que está haciendo la industria petrolera para resolverlo, además de presentar los parámetros a calcular en el procesador especifico. La sección III muestra una introducción a la metodología de diseño utilizada para el desarrollo del procesador específico. La secciones IV, V, VI, VII presentan el desarrollo del diseño del procesador específico para el cálculo del tiempo de viaje, la desaceleración lateral, el coseno del ángulo incidente y el ángulo emergente en el proceso de migración sísmica, por medio de la construcción del *Datapath* y una Máquina de Estados (*FSM*). La sección VIII muestra resultados del procesador diseñado y se presenta la validación del diseño por medio de pruebas de simulación aprovechando la herramienta de *software ISE 10.1.* Finalmente las conclusiones y las referencias cierran el artículo.

2. MIGRACIÓN SÍSMICA

Las empresas relacionadas con la industria del petróleo realizan múltiples procesos para obtener las imágenes del subsuelo y uno de estos es la migración sísmica, en el cual se usan clústers de computadores para poder obtener resultados más rápidamente, *"debido al gran costo computacional ocasionado por los grandes volúmenes de datos que se deben procesar"*[1]. El problema principal de estas empresas es que el proceso de migración puede tardar meses, debido a que los volúmenes de datos a procesar son del orden de Gigabytes [1].
La industria del petróleo, siempre está buscando reducir los tiempos de procesamiento para poder aumentar sus exploraciones anuales. "*Una muestra de ello son sus considerables inversiones económicas en clústers cada vez más grandes, que a su vez requieren de grandes sistemas de enfriamiento y su uso acarrea consumos considerables de energía"* [1], por lo tanto, "*la idea de tener un sistema de procesamiento específico que ayude a acelerar este proceso sin ocupar mucho espacio y cuyo consumo de potencia sea bajo, empezó a tener acogida dentro de la industria del petróleo"*[1], además, el crecimiento de los procesadores específicos, ha estado vinculado con el desarrollo de los FPGAs, en los que se han obtenido índices de aceleración buenos, comparados con los procesadores de propósito general.

Actualmente se encuentran dos clases de programas que son usados en el proceso de migración. La primera clase es de tipo industrial en la que el código fuente es privado porque son desarrollos de software que algunas empresas venden e incluso patentan. La segunda clase es de tipo académico, luego el código fuente es de libre uso y ha sido desarrollado en lugares como la escuela de minas de Colorado (Estados Unidos) [1].

Al revisar el código fuente que es de libre uso, se encuentra que matemáticamente la migración implica operaciones como suma, resta, multiplicación, división, potenciación, radicación, logaritmación y funciones trigonométricas. Esto se puede apreciar en las ecuaciones (2-9), que muestran las operaciones necesarias para calcular el tiempo de vuelo, la desaceleración lateral, el coseno del ángulo incidente y el ángulo emergente [1]. Cuatro parámetros necesarios dentro del proceso de migración sísmica que desde el punto de vista del formato de los datos, trabajan en punto flotante de precisión sencilla y además tienen un alto costo computacional [1].

2.1. Parámetros a calcular

Ahora entrando en más detalle, para el cálculo de estos parámetros se necesita del siguiente valor.

$$r_{ou} = \sqrt{r^2 + z^2} \tag{1}$$

Adicionalmente, los parámetros dependen de la variación de la velocidad en función de la profundidad para hacer sus cálculos, por esto, se cuenta con dos grupos de ecuaciones, las cuales se presentan a continuación. En primer lugar se presentan las ecuaciones de los parámetros para cuando no hay variación de la velocidad en función de la profundidad (*dVz = 0)*.

Tiempo de viaje:

$$t = \frac{r_{ou}}{V_o} \tag{2}$$

Desaceleración lateral:

$$P = \frac{r}{r_{ou}+V_o} \tag{3}$$

Coseno del ángulo incidente:

$$\cos\theta = \frac{z}{r_{ou}} \tag{4}$$

Ángulo emergente:

$$ang = \sin^{-1}\left(\frac{r}{r_{ou}}\right) \tag{5}$$

Mientras que el segundo grupo de ecuaciones (cuando la velocidad varía con la profundidad) son:

Tiempo de viaje:

(6)

Desaceleración lateral:

(7)

Coseno del ángulo incidente:

(8)

Ángulo emergente:

(9)

Resumiendo podemos decir, que con este trabajo se quiere realizar el diseño en lenguaje de descripción de hardware VHDL, de un procesador específico que calcule los parámetros mencionados y muestre una forma de potencializar la capacidad de procesamiento.

3. METODOLOGÍA DE DISEÑO

Un diseño digital utilizando un lenguaje de descripción de hardware es un proceso que puede ser tan complejo como se desee. En esta sección se pretende hacer una introducción a la metodología de diseño utilizada para la descripción en VHDL del procesador específico, que calcula los cuatro parámetros de la migración sísmica mencionados en la sección anterior, con un enfoque funcional, es decir, que el diseño funcione correctamente sin buscar alguna optimización en una métrica específica (velocidad, área, consumo de potencia, etc). La estrategia de trabajo estuvo basada en la metodología de diseño expuesta en [2], la cual consiste en diseñar módulos de *hardware* que cumplan una función específica, mediante la implementación de un *Datapath* controlado por una máquina de estados, este tipo de implementaciones se conocen en la literatura técnica como *FSMD* por sus siglas en inglés (Finite State Machine with Datapath) [2-3].

Un diseño digital parte de la definición de un objetivo a cumplir, es decir, las tareas específicas que el diseño debe realizar en el tiempo, en este caso el objetivo del procesador es calcular cuatro parámetros necesarios dentro del proceso de migración sísmica. Es por ello que en la Fig.1 se puede observar en el diagrama de flujo para el cálculo de los cuatro parámetros.

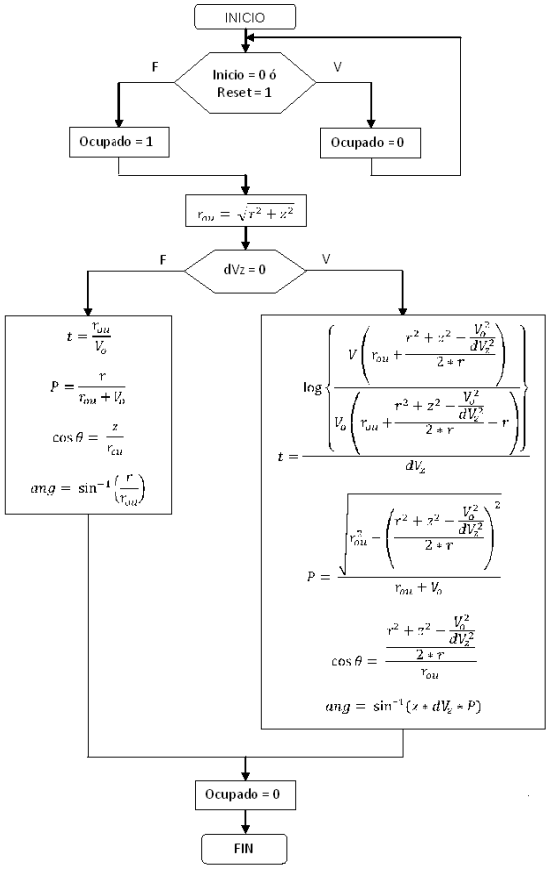


Fig1. Diagrama de flujo del procesador especifico.

Una vez conocido el objetivo que debe cumplir el diseño, se especifican tanto las entradas como salidas que necesita el procesador para funcionar correctamente [2]. En la Fig 2., se muestra las entradas y salidas del procesador.

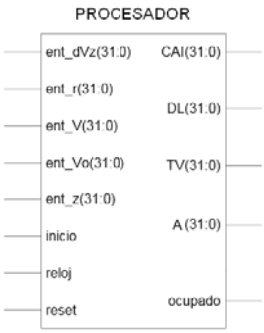


Fig 2. Entradas y salidas del Procesador Específico.

"Para poder implementar en hardware un algoritmo se tienen tres requerimientos: las variables, las unidades funcionales (para realizar la operaciones requeridas por el algoritmo) y la secuenciación en el tiempo de dichas operaciones" [2]. Un nivel de abstracción que ha resultado ser apropiado para representar un algoritmo secuencial es el Nivel de Transferencia de Registros (RTL), pues cuenta con *registros* que almacenan valores, *unidades funcionales* (sumador, restador, multiplicador, divisor, etc) para realizar las operaciones necesarias y un *control* que permite hacer la secuenciación en el tiempo requerido por el diseño, simulando así cada una de las iteraciones y los saltos de un algoritmo secuencial [2].

Por lo tanto, para realizar la implementación *FSMD* del procesador, se requiere de dos grandes bloques, el primero de ellos es el *Datapath* (sección V) que básicamente se encarga de ofrecer todo el *hardware* necesario (registros, sumadores, multiplexores, etc.) para poder cumplir con las necesidades del proceso y el Control (sección VI), el cual permite secuenciar las operaciones.

Un paso intermedio entre un algoritmo secuencial y una implementación *FSMD*, es la realización de un *Diagrama de Máquina de Estados Algorítmica* o un *Diagrama ASM* como se explica en [2]. El *Diagrama ASM,* contiene información más detallada acerca del algoritmo, lo cual facilita, tanto el diseño *Datapath* como el de la *FSM (Control)* y permite describir de manera gráfica una implementación *FSMD*.

Una vez construido el *Diagrama ASM*, diseñado el *Datapath* y definidas las palabras de control para la *FSM*, lo que se hace es realizar en VHDL la descripción de la implementación *FSMD.* El enfoque utilizado para describir la implementación FSMD en VHDL, se denomina multi-segmento, porque permite detallar la descripción, siguiendo un esquema de bloques y utilizando el estilo estructural de VHDL.

En las siguientes secciones se detallan cada uno de los pasos expuestos anteriormente, para el diseño del procesador específico que calcula los cuatro parámetros necesarios en la migración sísmica 2D.

4. DISEÑO DEL DIAGRAMA ASM

4.1. Selección de los registros y módulos funcionales:

Así como se mencionó en la sección anterior, uno de los pasos intermedios para facilitar el diseño del *Datapath y* de la *FSM*, es la realización de un *Diagrama ASM* [4], el cual permite de manera gráfica hacer una descripción más clara de las acciones que se deben realizar, para el correcto funcionamiento del procesador específico.

Antes de realizar el *Diagrama ASM* se debe conocer cuáles son y cómo es el funcionamiento de cada uno de los módulos necesarios para el desarrollo del procesador. Por lo tanto, observando las ecuaciones 1 a 9 de la sección II, se puede precisar que los módulos necesarios para la creación del procesador son:

Registros
Comparador
Multiplexores
Operadores suma, resta, producto, división y raíz cuadrada en punto flotante.
Función logaritmo en punto flotante.
Función trigonométrica arcoseno en punto flotante.

- *Registros*

Las entradas de los registros son variables de 32 bits. En el diseño también se incluye una señal de control L con el fin de que el registro inicie su operación en el momento que esta señal este en alto (1), véase [5-7].

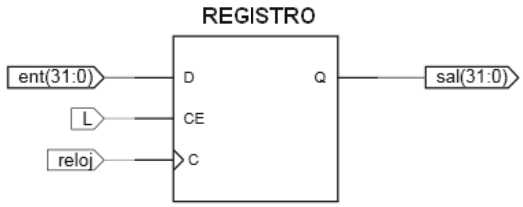


Fig 3. Diagrama de bloques Registro de 32 bits.

- *Comparador*

Para el diseño del procesador es necesario un comparador, véase [5-7], el cual permita contrastar la variación de la velocidad en función de la profundidad (*dVz*) con el valor cero y así poder escoger el grupo de ecuaciones a utilizar durante el cálculo de los parámetros.

Fig 4. Diagrama de bloques Comparador.

Operaciones aritméticas:

Suma, Resta, Multiplicación, División, Raíz cuadrada en punto flotante:

El CORE Generator de Xilinx es una herramienta flexible de alto rendimiento que ofrece a los diseñadores los medios

para realizar operaciones aritméticas basadas en FPGAs, además permiten generar el núcleo de la descripción de hardware para la función trigonométrica arco-tangente por medio de su módulo CORDIC de Xilinx [9-10].

Adicionalmente el operador de punto flotante del CORE Generator de Xilinx puede ser configurado para proporcionar una gama de operaciones y ofrece los núcleos de la descripción de hardware de la suma, resta, multiplicación, división, raíz cuadrada. La operación se especifica y cada variante tiene una interfaz común. Esta interfaz se muestra en la Fig 5. Cuando un usuario selecciona una operación que sólo requiere un operando, la entrada B es omitida que para este caso es la operación raíz cuadrada [9].

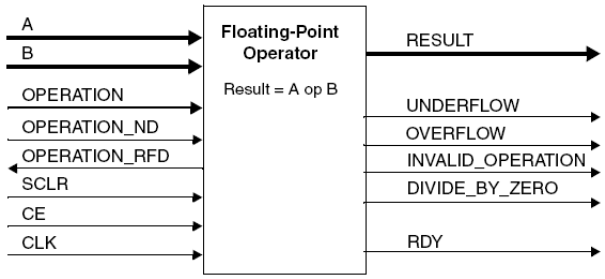


Figura 5. Diagrama de bloques del operador en punto flotante [9].

Los puertos utilizados en el diagrama de bloques que se muestra en la Fig 5 se describen en mayor detalle a continuación [9]:

ENTRADAS:

A: Entrada A de la operación. Número representado en punto flotante de 32 bits.

B: Entrada B de la operación. Número representado en punto flotante de 32 bits. En el caso de la raíz cuadrada esta entrada es deshabilitada.

CLK: Reloj. Todas las señales del módulo son sincrónicas[5] a la señal de reloj.

CE (Inicio): Cuando CE es deshabilitado (0), el reloj se desactiva, y el estado del módulo y sus resultados se mantienen.

SCLR (reset): Cuando SCLR se habilita (1), el control central reinicia de forma sincrónica el módulo a su estado inicial. Todos los resultados incompletos se descartan y RDY no se genera. Mientras SCLR se encuentre habilitado OPERATION_RFD y RDY son deshabilitados (0). El modulo está listo para un nuevo valor de entrada después de deshabilitar SCLR, y OPERATION_RFD se habilita (1).

OPERACIÓN: Especifica la operación a ser realizada. Las operaciones son números binarios codificados como se especifica en la tabla 1 [9].

Operación punto flotante		**Operación [5:0]**
Suma		000000
Resta		000001
Comparador (Programable)	Sin ordenar	000100
	Menor que	001100
	Igual	010100
	Menor igual	011100
	Mayor que	100100
	Diferente a	101100
	Mayor o igual	110100

Tabla 1. Codificación de operación [9].

OPERATION_ND (nd: nuevo dato): Debe estar en alto (1) para indicar que los operandos A, B y operación son validos, si OPERACION_ND está en bajo (0) impide la iniciación de nuevas operaciones y la posterior afirmación de RDY. Esta entrada es necesaria para sincronizar las operaciones cuando el módulo está configurado para ciclos de división y raíz cuadrada [9].

SALIDAS:

OPERACIÓN_RFD (rfd): Cuando está en alto (1), indica que el módulo se encuentra listo para leer los datos de entrada A, B y el código de la operación a realizar. Una nueva operación se inicia cuando *OPERATION_RFD y OPERATION_ND* están en alto.

RESULT: Esta señal muestra el resultado de la operación del modulo.

UNDERFLOW (u): Se establece en alto cuando la operación genera un resultado demasiado pequeño para poder ser representado en la precisión elegida, que para este caso es precisión simple.
OVERFLOW (o): Se establece en alto cuando la operación genera un resultado demasiado grande para poder ser representado en la precisión elegida.

INVALID_OP (i): Esta señal se establece en alto cuando la operación no es válida.

DIVIDED_BY_ZERO (dz): Se establece en alto cuando el número del denominador es igual 0.

[5] Sincrónica: Que ocurre al mismo tiempo, en este caso en el flanco de subida del reloj.

RDY (rdy): Cuando la señal de RDY se establece en alto, indica que el valor de la operación en ese momento es válido.

Función logaritmo:

El módulo para la función logaritmo fue tomado del trabajo *"A hardware-independent fast logarithm approximation with adjustable accuracy"* [11]. Este módulo permite calcular el valor de logaritmo natural de un número en punto flotante, pero para el cálculo del tiempo de viaje se necesita el logaritmo en base 10, por lo tanto se aprovecho este módulo y luego se hizo un cambio de base como se muestra en las ecuaciones (10) – (12).

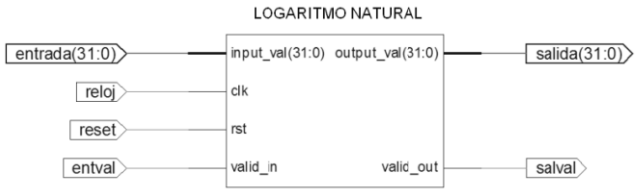


Fig 6. Diagrama de bloques logaritmo natural.

Función trigonométrica:

"CORDIC (COordinate Rotation DIgital Computer), o el método de dígito por dígito, o el algoritmo de Volder, es un simple y eficiente algoritmo para calcular funciones hiperbólicas y trigonométricas" [12]. El módulo CORDIC de Xilinx [10], cuenta con un módulo que permite calcular un ángulo a partir de la función trigonométrica arco-tangente, el cual tiene como entradas el seno del ángulo (entrada Y) y el coseno del ángulo (entrada X). Por lo tanto, para calcular el ángulo emergente en el proceso de migración sísmica, ecuaciones (5) y (9), se calculó el seno del ángulo y haciendo referencia a las identidades trigonométricas ecuaciones (13) y (14) también se calculó el coseno del ángulo, para poder utilizar el módulo de CORDIC y así hallar el valor del ángulo utilizando la ecuación (15).

Por otro lado, CORDIC es un módulo donde sus entradas y salidas trabajan en punto fijo, por lo tanto, para este módulo es necesario convertir las entradas de punto flotante a punto fijo y las salidas de punto fijo a punto flotante utilizando módulos que hagan este tipo de conversiones (ver Fig 7). Los módulos de conversión implican introducir un error en el cálculo de los resultados del valor del ángulo, para el caso de este diseño la aproximación en los resultados es de cuatro cifras decimales.

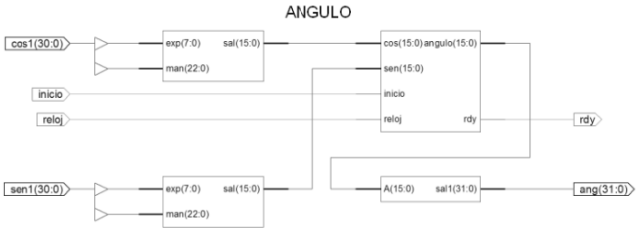
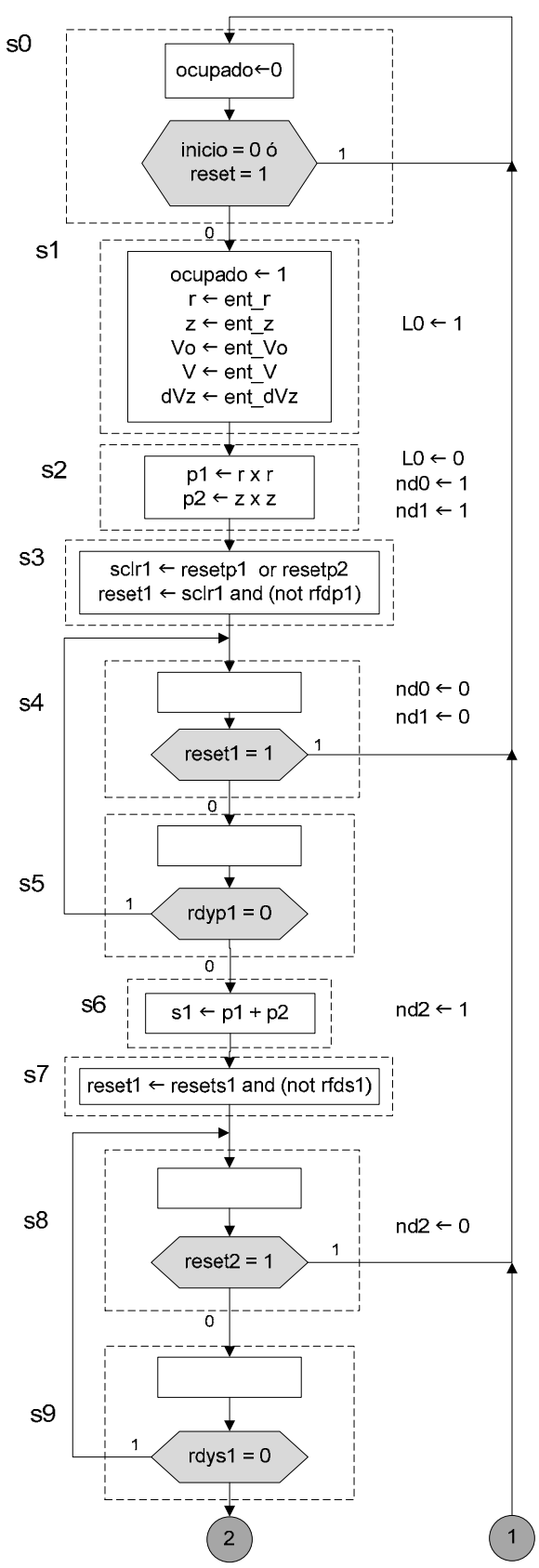


Fig 7. Diagrama de bloques del módulo arcotangente.

4.2. Diagrama ASM

Ya conocidos los módulos a utilizar en el diseño, se procedió a realizar el *Diagrama ASM.* En la Fig 8. se muestra una parte del *Diagrama ASM* que se utilizó en el diseño del procesador específico. Revisando el diagrama de la Fig. 8 se puede comprobar que éste permite el cálculo del valor r_{ou}, valor necesario en el cálculo de los cuatro parámetros que se desean desarrollar.

El primer estado es s0, este estado es utilizado para alertar al procesador del inicio del procesamiento de los datos. En el estado s1 se cargan los valores necesarios para el cálculo de los parámetros. Los estados donde se presentan cajas vacías simbolizan estados de espera, necesarios mientras se espera los resultados válidos de las operaciones (módulos funcionales). Los estados s1 a s14 describen como se calcula el valor $r_{ou.}$



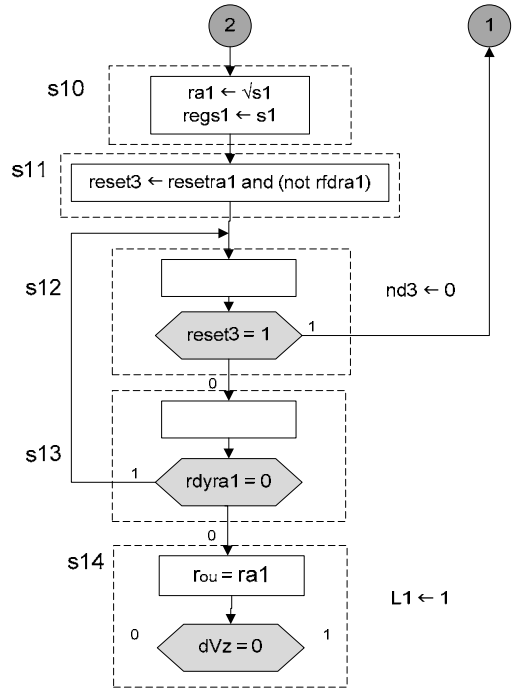


Fig 8. Diagrama ASM para el cálculo del valor r_{ou}.

5. DISEÑO DEL DATAPATH

El diseño de un *Datapath* puede ser tan variado como el de un algoritmo en software, es decir, no hay soluciones únicas. Para el diseño del *Datapath* se realizaron los siguientes pasos:

- *Determinar el número de registros a utilizar:*

Una tarea muy importante al diseñar un *Datapath* es determinar el número de registros a utilizar, por medio del *Diagrama ASM* se encontró que para este caso los registros a utilizar son 17, los cuales se enumeran a continuación:

- 5 registros para los valores de entrada del procesador (r, z, V, Vo, dVz).
- 6 registros para los valores de salida de operaciones (rp1, rp2, rs1, rd1, rd2, rd3).
- 1 registro para el valor de rou.
- 1 registro para el valor de seno del ángulo emergente (sen).
- 4 registros para los valores de salida del procesador (TV, DL, CAI, AE).

- *Enumerar las operaciones RT y agruparlas de acuerdo con su registro destino* [2].

Operaciones con el registro r:

- $r \leftarrow ent\_r$, en S_1.

Operaciones con el registro z:

- $z \leftarrow ent\_z$, en S_1.

Operaciones con el registro V:

- $V \leftarrow ent\_V$, en S_1.

Operaciones con el registro Vo:

- $Vo \leftarrow ent\_Vo$, en S_1.

Operaciones con el registro dVz:

- $dVz \leftarrow ent\_dVz$, en S_1.

Operaciones con el registro $rp1$:

- $rp1 \leftarrow p1$, en $S_{61,}$ $S_{69,}$ S_{77}.

Operaciones con el registro $rp2$:

- $rp2 \leftarrow p2$, en $S_{77.}$

Operaciones con el registro $rs1$:

- $rs1 \leftarrow s1$, en $S_{10.}$

Operaciones con el registro $rd1$:

- $rd1 \leftarrow d1$, en $S_{69.}$

Operaciones con el registro $rd2$:

- $rd2 \leftarrow d2$, en S_{69}.

Operaciones con el registro $rd3$:

- $rd3 \leftarrow d3$, en S_{53}.

Operaciones con el registro r_{ou}:

- $r_{ou} \leftarrow ra1$, en S_{14}.

Operaciones con el registro sen:

- $sen \leftarrow d1$, en S_{19}.
- $sen \leftarrow p1$, en S_{73}.

Operaciones con el registro TV:

- $TV \leftarrow d1$, en S_{36}, S_{89}.

Operaciones con el registro DL:

- $DL \leftarrow d2$, en S_{36}.
- $DL \leftarrow rd2$, en S_{89}.

Operaciones con el registro CAI:

- $CAI \leftarrow d3$, en S_{36}.
- $CAI \leftarrow d2$, en S_{89}.

Operaciones con el registro AE:

- $AE \leftarrow ang1$, en S_{36}, S_{89}.

Al igual que los registros, es muy importante al diseñar determinar el número de módulos funcionales, y enumerar las operaciones con cada una de las entradas, agrupándolas de acuerdo con su destino. Pero en este caso los valores de entrada no deben ser cambiados hasta que uno de los estados lo decida.

- 2 módulos de producto (p1, p2).
- 1 módulo de suma (s1).
- 1 modulo de raíz (ra1).
- 2 módulos de resta (re1, re2).
- 3 módulos de división (d1,d2,d3).
- 1 módulo de logaritmo natural (LN).
- 1 módulo de ángulo (ang1).

Producto 1(p1):

Operaciones con el producto 1 entradas A y B*:*

- $A \leftarrow r$, en S_2. $B \leftarrow r$, en S_2.
- $A \leftarrow sen$, en S_{23}. $B \leftarrow sen$, en S_{23}.
- $A \leftarrow Vo$, en S_{37}. $B \leftarrow Vo$, en S_{37}.
- $A \leftarrow d3$, en S_{53}. $B \leftarrow d3$, en S_{53}.
- $A \leftarrow s1$, en S_{57}. $B \leftarrow V$, en S_{57}.
- $A \leftarrow z$, en S_{65}. $B \leftarrow dVz$, en S_{65}.
- $A \leftarrow rp1$, en S_{69a}. $B \leftarrow rd2$, en S_{69a}.
- $A \leftarrow LN$, en S_{73}. $B \leftarrow K$, en S_{73}.

Producto 2(p2):

Operaciones con el producto 2 entradas A y B*:*

- $A \leftarrow z$, en S_2. $B \leftarrow z$, en S_2.
- $A \leftarrow dVz$, en S_{23}. $B \leftarrow dVz$, en S_{23}.
- $A \leftarrow r_{ou}$, en S_{37}. $B \leftarrow r_{ou}$, en S_{37}.
- $A \leftarrow re1$, en S_{53}. $B \leftarrow Vo$, en S_{53}.
- $A \leftarrow p1$, en S_{57}. $B \leftarrow p1$, en S_{57}.

Sumador 1(s1):

Operaciones con el sumador 1 entradas A y B*:*

- $A \leftarrow p1$, en S_6. $B \leftarrow p2$, en S_6.
- $A \leftarrow Vo$, en S_{15}. $B \leftarrow r_{ou}$, en S_{15}.
- $A \leftarrow r$, en S_{45}. $B \leftarrow r$, en S_{45}.
- $A \leftarrow r_{ou}$, en S_{53}. $B \leftarrow d3$, en S_{53}.
- $A \leftarrow r_{ou}$, en S_{61}. $B \leftarrow Vo$, en S_{61}.

Raíz cuadrada 1(ra1):

Operaciones con la raíz cuadrada 1 entrada A*:*

- $A \leftarrow s1$, en S_{10}.
- $A \leftarrow re1$, en S_{31}.
- $A \leftarrow re2$, en $S_{61,}$ S_{81}.

Restador 1(re1):

Operaciones con el restador 1 entradas A y B*:*

- $A \leftarrow 1$, en S_{27}. $B \leftarrow p1$, en S_{27}.
- $A \leftarrow s1$, en S_{45}. $B \leftarrow d2$, en S_{45}.
- $A \leftarrow s1$, en S_{57}. $B \leftarrow r$, en S_{57}.

Restador 2(re2):

Operaciones con el restador 2 entradas A y B*:*

- $A \leftarrow p1$, en S_{57}. $B \leftarrow p2$, en S_{57}.

- , en S_{77a}. , en S_{77a}.

Divisor 1(d1):

Operaciones con el divisor 1 entradas A y B*:*

- , en S_{15}. , en S_{15}.
- , en S_{19}. , en S_{19}.
- , en S_{65}. , en S_{65}.
- , en S_{85}. , en S_{85}.

Divisor 2(d2):

Operaciones con el divisor 2 entradas A y B*:*

- , en S_{19}. , en S_{19}.
- , en S_{41}. , en S_{41}.
- , en S_{65}. , en S_{65}.
- , en S_{85} , en S_{85}.

Divisor 3(d3):

Operaciones con el divisor 3 entradas A y B*:*

- , en S_{19}. , en S_{19}.
- , en S_{49}. , en S_{49}.
- , en S_{85}. , en S_{85}.

Después de haber realizado la agrupación de las operaciones de acuerdo a su registro destino, se puede utilizar esta información y conociendo los módulos funcionales, para obtener la cantidad de multiplexores necesarios en el diseño del *Datapath,* incluyendo el número de entradas de cada uno de los multiplexores. De acuerdo a lo anterior se puede concluir que el número de multiplexores en el presente diseño es el siguiente:

- 7 multiplexores 2 a 1:
  - 1 para el registro *sen.*
  - 1 para el registro *DL.*
  - 1 para el registro *CAI.*
  - 2 para las entradas A y B de *re2.*
  - 2 para las entradas A y B de *d3.*

- 7 multiplexores 4 a 1:
  - 1 para las entradas A de *ra1.*
  - 2 para las entradas A y B de *re1.*
  - 2 para las entradas A y B de *d1.*
  - 2 para las entradas A y B de *d2.*

- 6 multiplexores 8 a 1:
  - 2 para las entradas A y B de *p1.*
  - 2 para las entradas A y B de *p2.*
  - 2 para las entradas A y B de *s1.*

Continuando con el procedimiento, el siguiente paso fue la construcción de los circuitos para cada registro y módulo funcional, adicionando los multiplexores a cada uno de ellos, para direccionar los datos, el comparador, la función logaritmo y la función trigonométrica. En la Fig 9. se puede observar la construcción del circuito para el módulo funcional divisor 3, donde al módulo funcional se le adiciona en las entradas A y B dos multiplexores 4 a 1. Luego se interconectan los diferentes circuitos para así completar el *Datapath.*

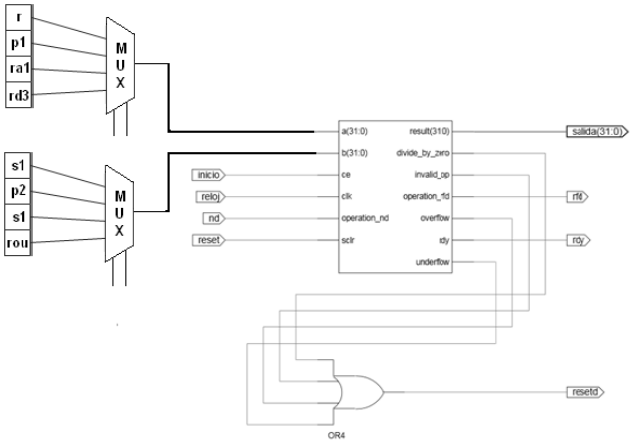


Fig 9. Circuito relacionado con el módulo funcional divisor 3 (d3).

6. DISEÑO DE LA FSM

"La FSM, es una máquina de estados finitos, que examina los comandos externos y las señales de estado para generar palabras de control que le permitan al Datapath realizar las operaciones RT en el orden establecido" [2].

Generalmente una FSM incluye las siguientes señales:

- **Comandos:** son señales de entrada a la FSM, que le permiten tomar decisiones, como por ejemplo el *inicio* de una operación.
- **Señales de Estado:** son señales entrada, que provienen desde el *Datapath*, las cuales son usadas junto con las señales de comando para determinar el siguiente estado.
- **Señales de control:** son señales de salida de la FSM y con usadas para controlar las operaciones que se realizan en el *Datapath.*
- **Salidas de control:** son señales de salida de la FSM que permiten identificar el estado de la FSMD como por ejemplo *ocupado, realizado*, etc [6].

En algunos estados se puede observar que hay más de una operación, por lo tanto, se necesita saber cuál es el RDY (señales de estado) a escoger para decidir cuándo se pasa de un estado al otro. Por eso, para esta decisión se toma el RDY de la operación que tarda más en arrojar los resultados

válidos, en la *Tabla 2*, se presenta la duración en pulsos de reloj de las diferentes operaciones utilizadas en el diseño.

OPERACIÓN	DURACIÓN(# pulsos)
Producto	9
Sumador	12
Restador	12
Raíz	28
Divisor	28
Arco-tangente	20
Logaritmo natural	23

Tabla 2. Duración en pulsos de reloj de operaciones.

"La FSM genera una palabra de control para cada uno de los estados en el diagrama ASM" [2]. Por medio de esta palabra se controlan las señales de habilitación de escritura de los registros, se activa las señales de OPERATION_ND de los módulos de operaciones aritméticas y del módulo arco-tangente, la señal de habilitación del módulo logaritmo natural y se direccionan los datos por medio de las señales de control de los multiplexores [2].

Por ejemplo para realizar la operación RT: $rs1 \leftarrow s1$, en S_{10}, se requerirá enviarle al *Datapath* la siguiente palabra de control:

E	L (9 a 0)	nd(9 a 0)
0	0000001000	0000001000

M(20 a 0)	Estado
00010010000000001000	S27

Tabla 3. Palabra de control de la FSM.

En donde:

- *E:* habilita el módulo logaritmo natural.
- *Li:* habilitan los diferentes registros, por ejemplo *L8* habilita el registro rd3.
- *ndi:* Indica que los valores en la entrada de los módulos funcionales son válidos, por ejemplo cuando *nd2* está en alto, el *sumador1* lee los datos a la entrada y comienza a realizar la operación válida para el diseño.
- *Mi:* señales de los diferentes multiplexores, por ejemplo *M (17 a 19)* son señales de los multiplexores M85 y M86.

7. DESCRIPCIÓN DEL PROCESADOR EN UN HDL

"El estilo de diseño que se propone para hacer la descripción en HDL del Datapath es estructural, es decir, cada una de las unidades funcionales se describe por aparte y luego se unen en un archivo de alto nivel" [2]. De otro lado para la descripción de la FSM se utilizó el estilo multi-segmento [2-3].

Por último, se realiza la implementación *FSMD* en VHDL (o en cualquier otro lenguaje HDL) de una manera sistemática, uniendo los archivos del *Datapath* y la *FSM* en un archivo de alto nivel.

RESULTADOS

La verificación del diseño se realizó mediante pruebas de simulación. En la Fig 10. se muestra la simulación del procesador específico para el cálculo de los cuatro parámetros utilizando el software *ISE Simulator (ISim)* de *Xilinx* versión 10.1.

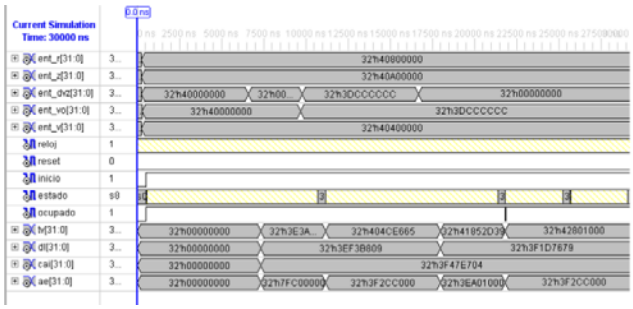


Fig 10. Simulación de la implementación FSMD del procesador específico.

En la fig 10 se puede observar: las primeras 5 filas representan las entradas que son *r, z, dVz, Vo, V*, las siguientes filas son el reloj del sistema, la señal de *reset*, la señal de inicio con la cual se da comienzo o enciende el procesador, la señal de estado donde se indica en cual estado de la *FSM* se encuentra el procesador en cada instante de tiempo, una señal de ocupado que indica si el procesador se encuentra realizando operaciones (1) o si está listo para recibir los siguientes datos de entrada (0). Por último, se encuentran 4 filas que nos arrojan las salidas del sistema que son: tiempo de viaje (TV), desaceleración lateral (DL), coseno del ángulo incidente (CAI) y el ángulo emergente (AE).

En la *Tabla 4.* se presentan los resultados de salida del procesador cuando los datos de entrada son: $r = 4,\ z = 5,$ $Vo = 0.1\ y\ V = 3$

Parámetro	$dVz \neq 0$	$dVz = 0$
TV	16.64708	64.03125
DL	0.61508	0.61508
CAI	0.78086	0.78086
AE	0.31262	0.67480

Tabla 4. Datos de salida del procesador.

En las *Tabla 5 y 6.* se presenta el cálculo de los porcentajes de error para cada uno de los parámetros, con los mismos valores de entrada con que se calcularon en la *Tabla 4*.

Parámetro	*Teórico*	*Simulación*	*%Error*
TV	16.64730	16.64708	0.00132
DL	0.61508	0.61508	0
CAI	0.78086	0.78086	0
AE	0.31261	0.31262	0.00319

Tabla 5. Porcentajes de error para valores de salida cuando dVz ≠ 0.

Parámetro	*Teórico*	*Simulación*	*%Error*
TV	64.03123	64.03125	0.00003
DL	0.61508	0.61508	0
CAI	0.78086	0.78086	0
AE	0.67474	0.67480	0.00889

Tabla 6. Porcentajes de error para valores de salida cuando dVz = 0.

CONCLUSIÓN

En este trabajo se presentó el diseño de un procesador específico que calcula cuatro parámetros necesarios para el proceso de migración sísmica, basado en una metodología de diseño con un enfoque funcional, sin buscar alguna optimización en velocidad, área o consumo de potencia. Del proceso de diseño se pudo concluir que:

- Para el desarrollo del diseño, un paso de gran importancia fue la realización del *Diagrama ASM,* el cual permitió de manera gráfica hacer una descripción más clara y sencilla de cada uno de los pasos y las acciones que se debían realizar para el correcto funcionamiento del procesador específico, disminuyendo notablemente los tiempos de diseño.

- Los *Diagramas ASM* permiten paralelizar operaciones a diferencia de los diagramas de flujo convencionales, lo cual permite hacer diseños en los que se busque paralelizar la mayor cantidad de operaciones con el fin de ganar velocidad de procesamiento.

- Una ventaja que ofrece esta metodología de diseño, es que facilita la construcción del *Datapath,* la definición de las palabras de control para la *FSM*, lo que causa realizar la implementación *FSMD* en VHDL de una manera sistemática.

- Por medio de pruebas de simulación realizadas en la herramienta de software *ISE 10.1*, se verificó el correcto funcionamiento del procesador específico, encontrando en los datos de salida del procesador unos porcentajes de error con respecto a valores calculados teóricamente (Matlab) del orden 1 x 10^{-3} %.

BIBLIOGRAFÍA

[1] S. A. ABREO. *"Viabilidad de acelerar la migración sísmica 2D pre-apilada en tiempo, usando un procesador específico implementado sobre un FPGA*", Articulo del estado del arte, seminario II, february 11, 2009.

[2] FAJARDO A. Carlos. RAMÓN Jorge H. *"Descripción de una metodología para diseñar procesadores de propósito específico implementados sobre FPGAs"*. XV Simposio De Tratamiento de Señales, Imágenes y Visión Artificial - STSIVA 2010.

[3] CHU, Pong P. *"FPGA PROTOTYPING BY VHDL EXAMPLES"*. Xilinx SpartanTM. 3. Pages 17-19. John Wiley & Sons. 2008.

[4] GAJSKI, Daniel. *"Principios de Diseño Digital".* Pearson 1997.

[5] Maxinez David G. Alcalá Jessica. *"VHDL el arte de programar sistemas digitales"*. TEC de Monterrey. CECSA. 2003.

[6] Thomas L. Floyd. **"***Digital Fundamentals with PLD Programming"*. Prentice Hall.2006.

[7] PARDO, Fernando. *"VHDL Lenguaje para descripcion y modelado de circuitos"*, Universidad de Valencia. (Octubre 14, 1997). http://www.fceia.unr.edu.ar/eca1/files/LDD/VHDL-Carpio.pdf.

[8] DI SERIO, Ángela. Organización del computador. Punto Flotante. CI-3815. Páginas 1-9. http://www.ldc.usb.ve/~adiserio/ci3815/clases/AritmeticaPuntoFlotante.pdf. Revisado en Junio 2010.

[9] XILINX. Logicore. "*Floating-Point Operator v3.0".* DS335 September 28, 2006. www.xilinx.com. Revisado en Junio 2010.

[10] XILINX. Logicore. "*CORDIC v3.0".* DS249 May 21, 2004. www.xilinx.com.

[11] ORIOL VINYALS, Gerald Friedland. *"A hardware-independent fast logarithm approximation with adjustable accuracy".* International Computer Science Institute 1947 Center Street, Suite 600 Berkeley, CA 94707 [vinyals,fractor]@icsi.berkeley.edu. Tenth IEEE International Symposium on Multimedia. http://opencores.org/project,fp_log.

[12] http://es.wikipedia.org/wiki/CORDIC. Revisado en Junio 2010.

AUTORES

Henyer Ricardo Sepúlveda Suárez, actualmente es estudiante de Ingeniería Electrónica de la Universidad Industrial de Santander. Auditor Interno de HSEQ. Sus intereses de investigación son el diseño de procesadores específicos y los FPGAs.

Carlos A. Fajardo Ariza, es Ingeniero Electrónico y Especialista en Docencia de la Universidad Industrial de Santander, en la cual es catedrático desde el año 2002. Actualmente se encuentra adscrito al grupo de investigación *Conectividad y procesamiento de Señales* (CPS) y adelanta estudios de Maestría en Ingeniería Electrónica en esta misma universidad. Sus campos de interés son los FPGAs, los Embedded System y la aceleración de algoritmos por medio de FPGAs.

Sergio Alberto Abreo Carrillo, nació en Bucaramanga, Colombia en el año de 1983. Actualmente es candidato a magíster en la Escuela de Ingeniería Eléctrica Electrónica y Telecomunicaciones de la Universidad Industrial de Santander. Además es profesor cátedra de la materia digitales en la misma institución y pertenece a los grupos de investigación en Conectividad y Procesamiento de Señales (CPS) y Petrosísmica. Se graduó como Ingeniero en Control Electrónico de la Universidad Distrital Francisco José de Caldas en Abril del 2008. Sus intereses de investigación incluyen diseño e implementación de procesadores específicos usando FPGAs, procesamiento digital de señales y computación de alto rendimiento. Actualmente participa en un proyecto de investigación que busca desarrollar un procesador específico aplicado a la sísmica exploratoria patrocinado por el convenio UIS-ICP 005 del 2003.